送信機をFPGAボードに実装して動かす

── 2005年1月号付属FPGA基板を使って動作を確認

第5章

横山明久

ここでは，QPSK（quadrature phase shift keying）変調方式の送信機，およびπ/4シフトQPSK変調方式とASK（amplitude shift keying）変調方式の切り替えが可能な送信機について，実際に回路データをFPGA上に展開して動作確認を行う．変調方式の切り替えには，外部のプッシュ・スイッチを利用している．なお，ここでは，本誌2005年1月号に付属したFPGA基板（米国Xilinx社の「Spartan-3 XC3S50-4VQ100」を搭載）にこれらの送信機を実装した．（編集部）

ここでは，第3章で解説したQPSK（quadrature phase shift keying）変調方式の送信機，および第4章で解説したπ/4シフトQPSK変調方式とASK（amplitude shift keying）変調方式の切り替えが可能な送信機をFPGA基板に実装して動作確認を行います．なお，ここでは論理合成，配置配線用ツールとして米国Xilinx社の「ISE」を使用しています注．

本稿では，まずFPGA基板の構成について説明します．次に，論理合成によってHDL記述をゲート・レベルの回路情報に変換するための設定方法を説明します．なお，論理合成，配置配線用ツールの使いかた，FPGA基板上のPROM（コンフィグレーションROM）用データの作成や書き込みの手順などについては，pp.99-104のコラム「論理合成，PROMデータ作成，PROM書き込み」を参照してください．

その後，第3章で設計したQPSK変調方式の送信機をFPGA基板に実装するために必要となる制約条件ファイル（ucfファイル）について説明します．そして，実機への実装を行い，ロジック・アナライザを利用して動作確認を行います．時系列データあるいは*IQ*コンスタレーションを作成し，シミュレーション結果と同じ出力が得られることを確認します．

また，第4章で設計したπ/4シフトQPSK変調方式とASK変調方式の切り替えが可能な送信機をFPGAに実装するために必要となる最上位HDLモジュールと制約条件ファイルについて説明します．搭載した`reset`信号と変調方式の切り替え信号を単一スイッチ信号から生成する回路についても説明します．実機への実装を行い，FPGA基板の`reset`ボタンのON/OFFによって変調方式が変化することを確認します．

注：筆者は今回，論理合成，配置配線ツールとして，本誌2005年1月号付属CD-ROMに収録されているWebPACK 6.3iを使用した．

1．実装に際しての前準備

今回は，回路データを実装するFPGAボードとして，本誌2005年1月号の付属基板を用います（**写真1**）．詳細については同号の記事を参照してください．

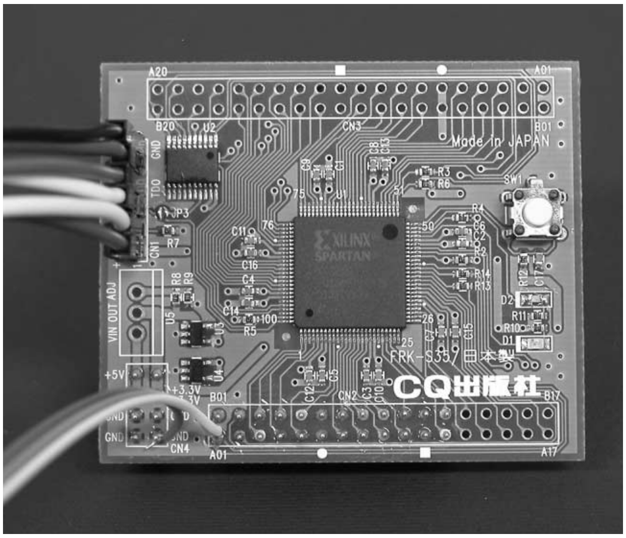


写真1　FPGA基板

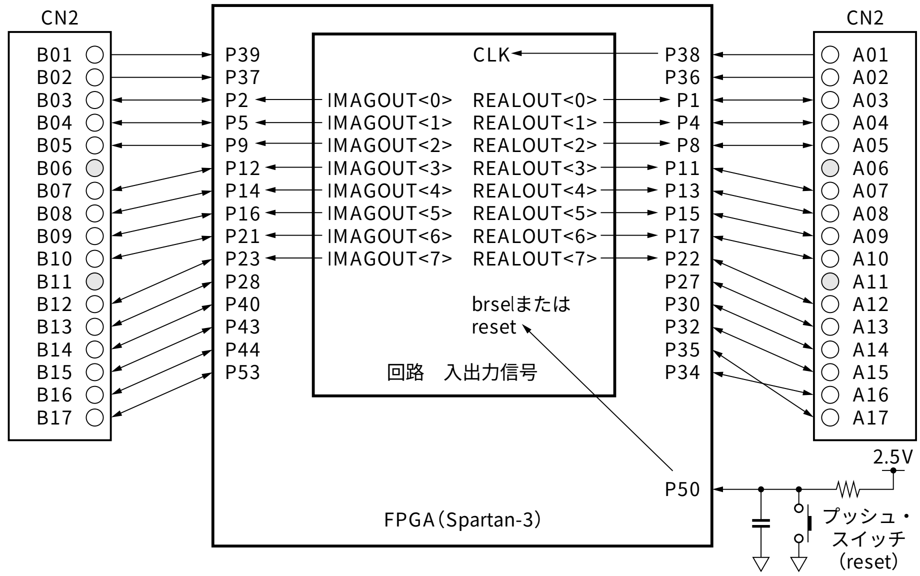


図1
FPGAボードの信号接続
FPGAボード上のコネクタやプッシュ・スイッチ，FPGA間の物理的な信号接続，およびFPGA内部の回路入出力信号と実際のピンの対応を示している．図の左右にある四角はユーザ拡張用I/O（コネクタCN2），中心の四角がFPGA，右下にあるプッシュ・ボタンはreset信号生成用である．

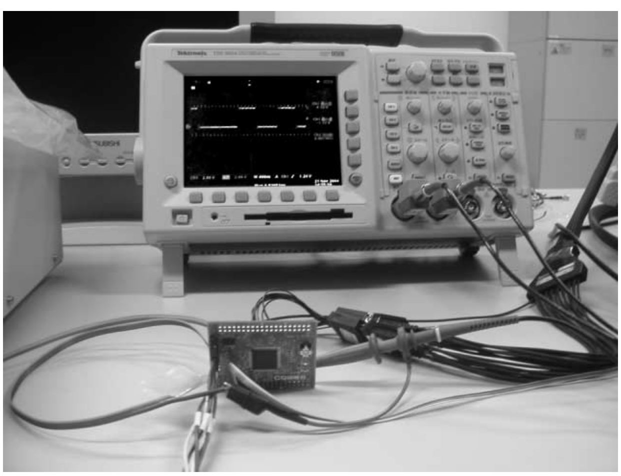


写真2　FPGA基板を動作させているようす

● 2005年1月号付属基板を活用

　この付属基板には，90nmルールのCMOSプロセスで製造された約5万ゲート相当である米国Xilinx社製のFPGA「Spartan-3（XC3S50-4VQ100）」が搭載されています．基板上にはレギュレータが搭載されており，3.3V単一電源の供給によって動作します．また，基板上には電源接続用パターンも用意されています．

　この基板はJTAGヘッダを備えているので，JTAG経由でコンフィグレーション・データをFPGAにダウンロードできます．また，FPGA専用のPROM（Xilinx社の「XCF01SVO20C」）用パターンが用意されています．PROMを搭載してそこにビット・ファイル・データを書き込むことで，電源投入時に自動的にコンフィグレーション・データをFPGAにロードすることができます．今回はPROMを搭載し，PROMデータの作成と書き込みを行って，FPGA基板単体で動作させることにします．

　本FPGA基板にはプッシュ・スイッチ（オムロンの「形B3W-1100」を推奨）を実装するパターンが用意されており，これによって外部からON/OFF信号を入力できます．今回はこのプッシュ・スイッチを搭載して`reset`信号を生成したり，変調信号の切り替え信号として利用しました．

　また，本FPGA基板はユーザ拡張用I/Oピンを備えています．クロック入力用のグローバル・クロック線が8系統，信号入出力用の汎用I/Oが50本となっています．**図1**に，ボード上のFPGAピンと汎用I/Oピンの間の接続を示します．**写真2**は，実際にFPGA基板を動作させたときのようすを撮影したものです．

　FPGA基板のユーザ端子から変調された*IQ*信号をディジタル値として出力するには，次のような手順をとります．

- **論理合成，配置配線後のビット・ファイルからROMデータを作成**
- **FPGA基板に3.3V電源を投入**
- **FPGAのコンフィグレーション，またはROMデータのPROM書き込み（コンフィグレーションにはダウンロード・ケーブル「Parallel Cable IV」を使用）**
- **8.192MHzのクロックをCLK端子へ入力（水晶発振子または信号発生器による）**
- **ロジック・アナライザによって出力信号を観測**

● 実装に必要なモジュール，ファイルを作成する

設計した回路を実際にFPGA上で動作させるためには，以下を作成，設定する必要があります．特記事項について3点を挙げます．

1）topモジュールの作成

論理合成時の最上位HDLモジュール（topモジュール）を作成します．topモジュールには，下位モジュール間の接続やFPGA外部とのインターフェースを記述します．

2）制約条件ファイルの作成

論理合成を実行するとき，ツールに指針を与えるのが制約条件ファイルです．回路規模や遅延時間，ピン割り当て（ピン・アサイン），動作周波数といった条件を与えます．ツールは，組み合わせ回路の構成を変更することで，ユーザの要求に合う回路の生成を試みます．もちろんツールは，どのような要求にでもこたえられるというわけではありません．FPGAの性能，あるいはソース・コードの記述方法のよしあしなどによって，その限界が決まります．また，一般に，回路規模と動作周波数はトレードオフの関係にあります．

ここでは，FPGAのピン割り当てについてのみ制約を設けました．今回は，Spartan-3を搭載したボードの拡張I/OピンA01～A08に出力信号REALOUTを，B01～B08に出力信号IMAGOUTをそれぞれ8ビット信号として割り当てています．また，クロックとプッシュ・スイッチ信号についても割り当てを行っています．**図1**は，回路のtopモジュールの入出力信号名とFPGAの物理ピン名の対応を示しています．**リスト1**に示す制約条件ファイルを見ると，**図1**に示した対応づけを行っていることがわかります．

3）乗算器LUT使用指示

今回使用したFPGA基板に搭載されているXC3S50には18ビットのハードウェア乗算器が搭載されています．しかし，このFPGAはハードウェア乗算器を四つしか内蔵していません．一方，今回設計したFIRフィルタではFIRフィルタ一つにつき9個の乗算器，合計18個の乗算器が必要になります．つまり，ハードウェア乗算器だけで実現することはできません．よって，FPGA内のルックアップ・テーブル（LUT）を利用して乗算器を構成する必要があります．

第4章の**リスト8**（p.86）に示しましたが，FIRフィルタの乗算処理を行う記述で，LUTを使用するように指定しています．LUTの使用の指定は，Verilog HDLの場合は，

リスト1　FPGAのピン割り当て

図1で示したFPGA内部の回路入出力信号と実際のピンの対応について記述した制約条件ファイルである．このファイルを論理合成ツールに入力する．クロック信号，変調方式の切り替え信号，*IQ*それぞれの8ビット信号といった入出力信号について記載してある．

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "gateway_in1"    LOC = "P38" ;
NET "gateway_in2"    LOC = "P50" ;
NET "converter1(0)"  LOC = "P1"  ;
NET "converter1(1)"  LOC = "P4"  ;
NET "converter1(2)"  LOC = "P8"  ;
NET "converter1(3)"  LOC = "P11" ;
NET "converter1(4)"  LOC = "P13" ;
NET "converter1(5)"  LOC = "P15" ;
NET "converter1(6)"  LOC = "P17" ;
NET "converter1(7)"  LOC = "P22" ;
NET "converter2(0)"  LOC = "P2"  ;
NET "converter2(1)"  LOC = "P5"  ;
NET "converter2(2)"  LOC = "P9"  ;
NET "converter2(3)"  LOC = "P12" ;
NET "converter2(4)"  LOC = "P14" ;
NET "converter2(5)"  LOC = "P16" ;
NET "converter2(6)"  LOC = "P21" ;
NET "converter2(7)"  LOC = "P23" ;


#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

```
// synthesis attribute mult_style of
  <モジュール名> is lut
```

という記述で行います．一方，VHDLの場合はモジュールへの属性設定を行うエンティティ宣言の，

```
attribute mult_style: string;
 attribute mult_style of <モジュール名> :
                         entity is "lut";
```

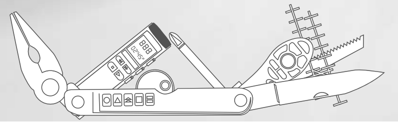
という記述で行います．

2．回路データの実装と動作検証

次に，設計した回路をFPGA上にて実際に動作させるまでの過程を説明します．

● QPSK変調方式の送信機をFPGAに実装

第3章では信号処理のシステム・レベル・シミュレータを利用して，HDLソース・コードの動作検証を行いました．このときに利用したXilinx社の「System Generator for DSP」は，論理合成に利用できるプロジェクト・ファイル（ファイル名はxst_simulation4qpsk_modulator2_clk_



wrapper.prj)を生成します．

System Generator for DSPを利用した場合，回路をFPGAに実装するために必要なことは，制約条件ファイルを作ることだけです．プロジェクトとして生成されたHDLファイルには，いっさい変更の必要はありません．制約条件ファイルについては，**図1**に示したFPGAの入出力ピン名とtopモジュールの入出力信号の対応を記述します．topモジュールで記述されたシステム内部ブロック図を**図2**に示します．

制約条件ファイルでは，topモジュールにあるすべての入出力信号に対して，実際のFPGAピンに対応させた制約を与えます(**リスト1**)．NETコマンドにより，入出力信号がビット単位で実際のピン名と結び付けられていることがわかります．この制約条件ファイルを利用して論理合成を行うと，実動作可能なビット・ファイルが生成されます．

ビット・ファイルの転送(pp.99-104のコラム「論理合成，PROMデータ作成，PROM書き込み」を参照)を行った後，CLK入力ピンに信号発生器から8.192MHzのクロックを入力し，FPGA上で動作する変調信号生成処理から出力される信号を観測しました．FPGA基板によるロジック・アナライザでの実動作の観測結果を**図3**に示します．*I*信号(`real_out`)と*Q*信号(`imag_out`)のそれぞれに対応した入出力ピンを経由して，変調信号が基板外部に出力されていることがわかります．

また，この信号データをロジック・アナライザを用いてファイル化し，(x，y)平面上にプロットして*IQ*コンスタレーションを確認したのが**図4**です．第3章で示したシミュレーション結果と同じになっていることがわかります．これで期待どおりの動作を行うことが確認できました．

● π/4シフトQPSKとASKに対応した送信機を実機検証

第4章で設計した回路データを利用して，変調方式をプッシュ・スイッチで切り替えられる送信機をFPGAに実装します．π/4シフトQPSK変調方式の場合と同じように，FPGA基板に実装した変調処理システムを動作させ，ロジック・アナライザで出力波形を観測します．*IQ*出力ピンにロジック・アナライザを接続し，CLK入力ピンに8.192 MHzクロックを入力させて，FPGAで動作する変調信号生成処理からの出力信号を観測します．

FPGAに実装した変調信号生成処理の構成を**図5**に示

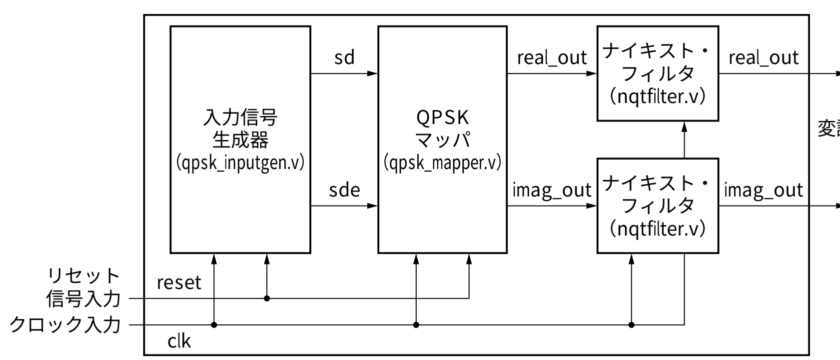


図2 topモジュールで記述されたシステム内部ブロック図

FPGA基板上に実装したQPSK変調信号生成処理システムの内部ブロック図．入力信号生成，*IQ*信号生成，ナイキスト・フィルタの3種類のブロックから構成されている．

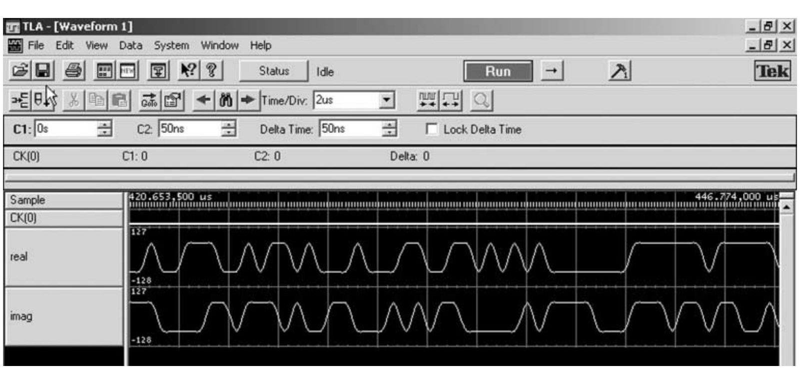


図3 変調信号生成処理から出力される信号の測定

基板上のコネクタに出力されるFPGAからの変調信号(ディジタル値)をロジック・アナライザで測定した結果．期待どおりの信号が出力されている．

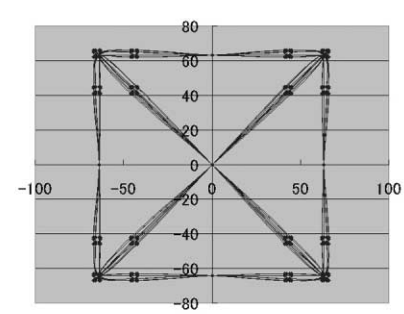


図4 QPSK変調出力データの*IQ*コンスタレーション

図3の測定で利用したロジック・アナライザを利用して*I*，*Q*それぞれの変調信号をファイル化し，(x，y)平面にプロットした．第3章の図2で示したように信号点空間における信号点位置を通過する期待どおりの結果となっている．

します．また，topモジュールのVerilog HDLソースを**リスト2**に示します(本誌付属のDVD-ROMにはVHDLソースも収録されている)．システム構成どおり，各下位モジュールが接続されていることがわかります．また，ピン割り当ての制約を**リスト3**に示します．基本的に**図1**に示すピン割り当てをそのまま記載してあります．

π/4シフトQPSK変調方式とASK変調方式の切り替えが可能な送信機を実装するにあたり，切り替え信号と変調方式の指示信号を生成する必要があります．ここでは，実装するFPGA基板のreset信号を利用して変調方式を切り替えることにします．さらに，PN系列生成部のシフト・レジスタを初期化するreset信号を生成するための簡単な論理回路を追加しています．

変調方式の切り替えやreset信号入力について，シミュレーション記述では初期化時にこれらを指示していますが，実機では外部からトリガとなる信号を与えてやらなければなりません．シフト・レジスタが初期化されないと，PN系列生成からの出力はつねに‘0’となってしまうため，ランダム値にならず，結果として変調信号は周期的な信号を生成することになるためです．変調処理の切り替え信号とreset信号は別の入力を用意する方法もありますが，今回は一つのスイッチで両方を制御できるようにしました．

その方法は以下のとおりです．reset信号生成回路では，スイッチのON/OFF状態をレジスタに格納しておき，

リスト2　topモジュール(top.v)

```
module top (clk, brsel, realout, imagout);

input         clk;
input         brsel;
output  [7:0] realout;
output  [7:0] imagout;


  wire        clk;
  wire        brsel;

  wire  [7:0] realout_ask;
  wire  [7:0] imagout_ask;
  wire        sd_ask_gen;
  wire        sde_ask_gen;
  wire        sd_ask;
  wire        sde_ask;

  wire  [7:0] realout_qpsk;
  wire  [7:0] imagout_qpsk;
  wire        sd_qpsk_gen;
  wire        sde_qpsk_gen;
  wire        sd_qpsk;
  wire        sde_qpsk;

  wire  [7:0] realout;
  wire  [7:0] imagout;

  wire  [7:0] realout_out;
  wire  [7:0] imagout_out;
  reg   [7:0] realout_reg;
  reg   [7:0] imagout_reg;

  reg         brsel_buf;
  reg         reset;

//********** main routine

  ask_inputgen s_ask_inputgen
    (.clk(clk), .reset(reset), .sd(sd_ask), .sde(sde_ask));

  qpsk_inputgen s_qpsk_inputgen
    (.clk(clk), .reset(reset), .sd(sd_qpsk), .sde(sde_qpsk));

  share_inputselect s_share_inputselect
    (.clk(clk),
     .rd_ask_in(sd_ask),    .rde_ask_in(sde_ask),
     .rd_qpsk_in(sd_qpsk), .rde_qpsk_in(sde_qpsk),
     .brsel(brsel),
     .sd_ask_out(sd_ask_gen), .sde_ask_out(sde_ask_gen),
     .sd_qpsk_out(sd_qpsk_gen), .sde_qpsk_out(sde_qpsk_gen)
    );

  ask_mapper s_ask_mapper
    (.clk(clk), .reset(1'b0),
     .rd(sd_ask_gen), .rde(sde_ask_gen),
     .real_out(realout_ask), .imag_out(imagout_ask)
    );

  qpsk_mapper s_qpsk_mapper
    (.clk(clk), .reset(1'b0),
     .rd(sd_qpsk_gen), .rde(sde_qpsk_gen),
     .real_out(realout_qpsk), .imag_out(imagout_qpsk)
    );

  share_modselect s_share_modselect
    (.clk(clk),
     .ask_real_in(realout_ask), .ask_imag_in(imagout_ask),
     .qpsk_real_in(realout_qpsk), .qpsk_imag_in(imagout_qpsk),
     .mod_real_out(realout_out), .mod_imag_out(imagout_out)
    );


always @(posedge clk) begin

//realout_reg[0] = sd_ask;
//realout_reg[1] = sde_ask;
//realout_reg[2] = sd_qpsk;
//realout_reg[3] = sde_qpsk;
//realout_reg[4] = sd_ask_gen;
//realout_reg[5] = sde_ask_gen;
//realout_reg[6] = sd_qpsk_gen;
//realout_reg[7] = sde_qpsk_gen;

//realout_reg    = realout_ask;
//imagout_reg    = realout_qpsk;

  realout_reg    = realout_out;
  imagout_reg    = imagout_out;

end

assign realout = realout_reg;
assign imagout = imagout_reg;

always @(posedge clk) begin
  if (brsel == brsel_buf) begin
    reset = 1'b0;
  end
  else begin
    reset = 1'b1;
  end
  brsel_buf = brsel;
end

endmodule
```

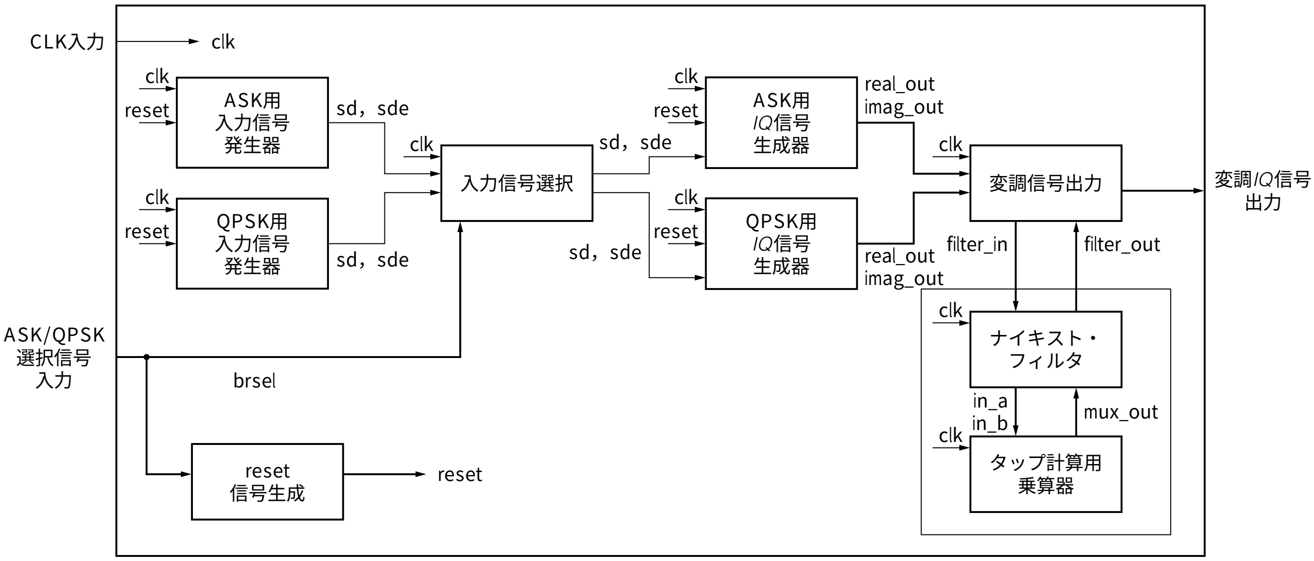


図5　変調信号生成処理システムの構成

ASK変調方式とπ/4シフトQPSK変調方式を切り替えられる送信機の中の変調信号生成処理に関するブロック図．ここにあるreset信号生成モジュールは第4章では作成していなかった．本章で新たに付け加えた回路である．

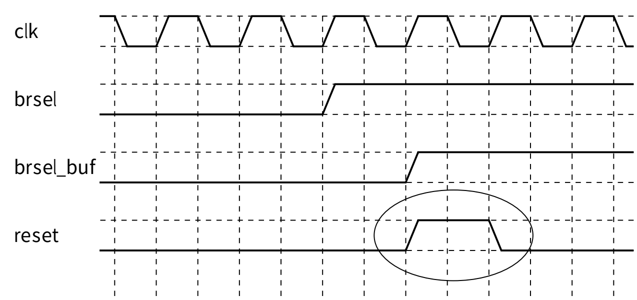


図6　reset信号生成のタイミング・チャート

`brsel`信号を利用した1クロック期間の`reset`信号処理をタイミング・チャートで示した．入力する`brsel`信号は，`brsel_buf`につねに退避させている．`brsel`信号と`brsel_buf`が異なる場合のみ，`reset`信号が'1'となる．

リスト3　ピン割り当て

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "CLK"          LOC = "P38" ;
NET "BRSEL"        LOC = "P50" ;
NET "REALOUT<0>"   LOC = "P1"  ;
NET "REALOUT<1>"   LOC = "P4"  ;
NET "REALOUT<2>"   LOC = "P8"  ;
NET "REALOUT<3>"   LOC = "P11" ;
NET "REALOUT<4>"   LOC = "P13" ;
NET "REALOUT<5>"   LOC = "P15" ;
NET "REALOUT<6>"   LOC = "P17" ;
NET "REALOUT<7>"   LOC = "P22" ;
NET "IMAGOUT<0>"   LOC = "P2"  ;
NET "IMAGOUT<1>"   LOC = "P5"  ;
NET "IMAGOUT<2>"   LOC = "P9"  ;
NET "IMAGOUT<3>"   LOC = "P12" ;
NET "IMAGOUT<4>"   LOC = "P14" ;
NET "IMAGOUT<5>"   LOC = "P16" ;
NET "IMAGOUT<6>"   LOC = "P21" ;
NET "IMAGOUT<7>"   LOC = "P23" ;

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

現在の状態と比較します．スイッチのON/OFF状態が変化したときに`reset`信号をONにします．`reset`信号は1クロック区間のみONとなります．その際，`reset`信号が接続された信号生成部においてシフト・レジスタが初期化されます．

図6に`reset`信号生成のタイミング・チャートを示します．1クロック区間のみ`reset`信号がONになっていることがわかると思います．

QPSK変調方式の送信機の場合と同様，topモジュールと制約条件ファイルを利用して，pp.99-104のコラムで紹介する論理合成，PROMデータ作成，PROM書き込みを行います．そして，信号発生器からCLK入力ピンに8.192MHzのクロックを入力し，送信機回路を動作させます．そのあと，FPGA上で動作する変調信号生成処理から出力される信号をロジック・アナライザで観測しました．

図7はASK変調信号の出力結果です．第4章のシミュレーション結果と同じく，正しくスプリット・フェーズ化された変調信号が出力されています．**図8**はπ/4シフトQPSK変調信号出力です．こちらは，*IQ*マッピング信号に対して，ナイキスト・フィルタを通過させることによって得られた信号です．それぞれの波形は第4章のシミュレーション結果と同じようになっています．

π/4シフトQPSK変調については，変調信号を測定した

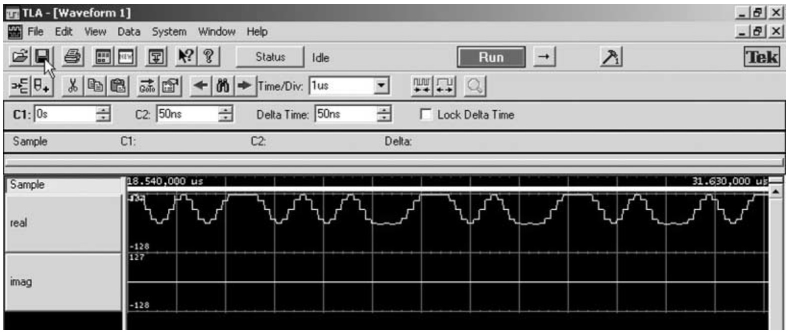


図7　ASK変調*IQ*出力信号の測定

ASK変調の出力信号がFPGA基板のコネクタに出力されているようすをロジック・アナライザで測定した．期待どおりの信号が出力されているようすがわかる．

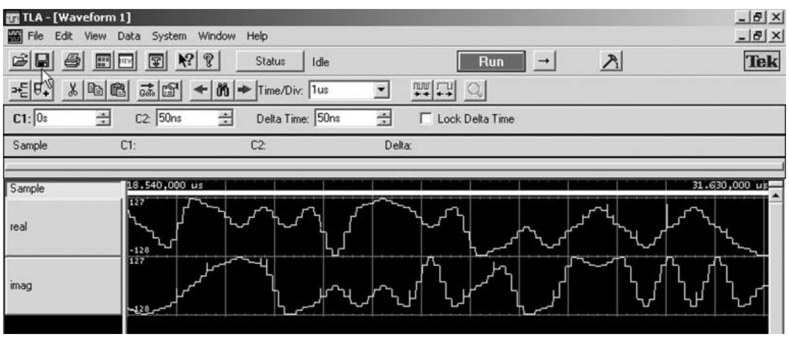


図8　π/4シフトQPSK変調*IQ*出力信号の測定

π/4シフトQPSK変調の出力信号がFPGA基板のコネクタに出力されているようすをロジック・アナライザで測定した．こちらも期待どおりの信号が出力されているようすがわかる．

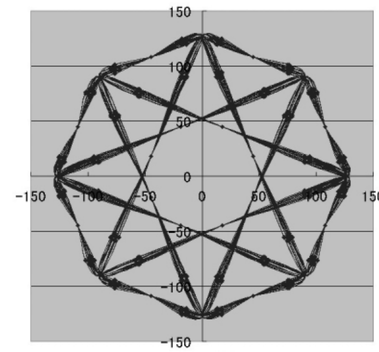


図9　π/4シフトQPSKの*IQ*コンスタレーション

図8の測定に利用したロジック・アナライザを使って*I*，*Q*それぞれの変調信号をファイル化し，(*x*，*y*)平面にプロットした．第4章の図6(p.84)で示した信号点空間における信号点位置を通過する期待どおりの結果となっている．

結果をロジック・アナライザを利用してファイルに出力し，*I*信号と*Q*信号のそれぞれの時系列データを(*x*，*y*)にプロットして*IQ*コンスタレーションを作図しました(**図9**)．タイミングが崩れることなく，正しく*IQ*コンスタレーションを出力できています．また，第4章のシミュレーション結果とも同じになっていることがわかります．実機上でもπ/4シフトQPSK変調処理が期待どおりに行われていることを確認できました．

このように，シミュレーションと同じ動作を実機で簡単に実現できるのがFPGAの魅力です．読者のみなさんも，今回のサンプル・コードを参考にして，変調処理をはじめとするさまざまな無線信号処理を動作させ，FPGAのおもしろさにぜひ触れていただきたいと思います．

よこやま・あきひさ
(株)トヨタIT開発センター

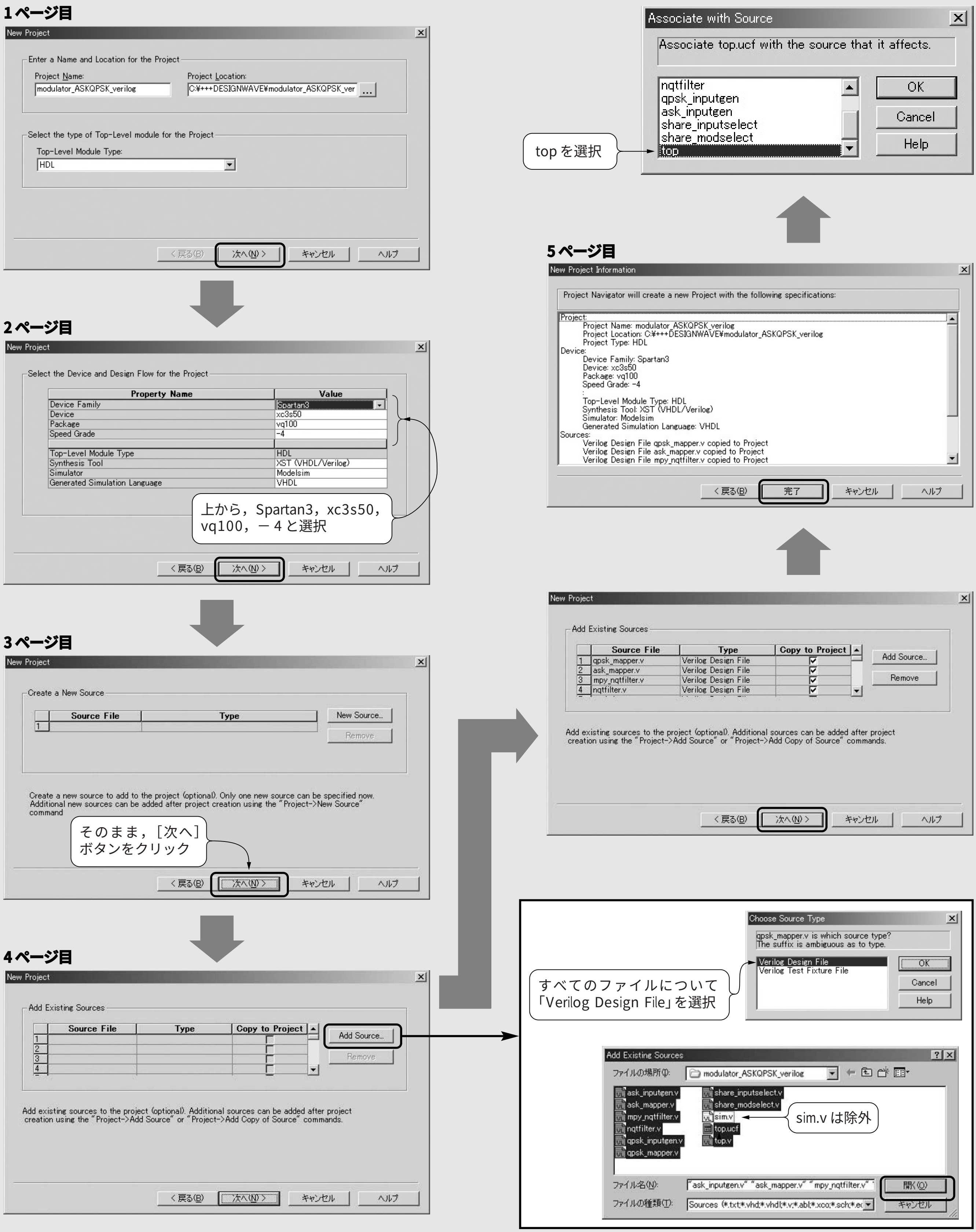
COLUMN　論理合成，PROMデータ作成，PROM書き込み

ここでは論理合成，PROM(コンフィグレーションROM)用データの作成，PROMの書き込みの過程について説明していきます．

● ファイルを指定して論理合成を行う

まず，米国Xilinx社の論理合成ツールである「XST(Xilinx Synthesis Technology)」を利用した合成手順を説明します(XSTは「ISE」に含まれている)．ここでは，Verilog HDLのソース・コードを例に説明しますが，VHDLの場合も手順は同じです(ファイル指定部分が異なるだけ)．

まず，Windowsのスタート・メニューから「プログラム」→「Xilinx ISE 6」→「Project Navigator」を選択して，ISEを起動します．次に，新しいプロジェクト・ファイルを作成するため，「File」→「New Project」を選択します(**図A-1**)．すると，**図A-2**の1ページ目のようなウィンドウが開きます．ここで，Project Location 欄にはHDLフ



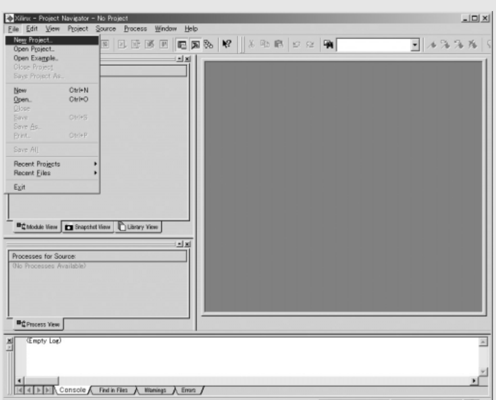
図A-2　新しいプロジェクトを生成する

ァイルが格納されたフォルダが入っているパスを選択します．HDLファイルがあるフォルダ名を「Project Name」とし（**図A-2**の例ではmodulator_ASKQPSK_verilogとなっている），［次へ］ボタンを押します．

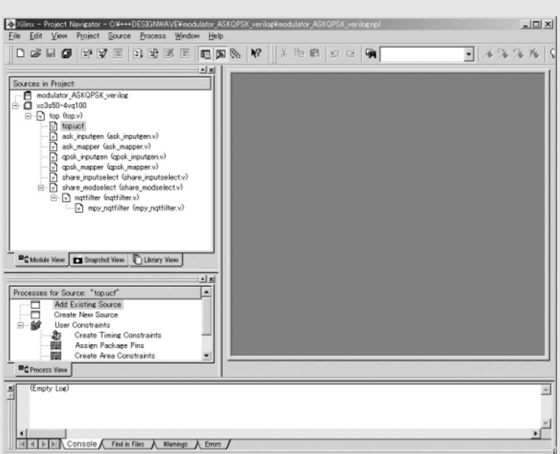
2ページ目のウィンドウで合成対象となるFPGAの種類を指定します．今回は，Device Family欄を「Spartan3」，Device欄を「xc3s50」，Package欄を「vq100」とします．3ページ目は新しいソース・ファイルを作成する場合に利用しますが，今回はすでに作成しているファイルを使うので，そのまま［次へ］ボタンを押します．

4ページ目では，すでに作成したソース・ファイルを指定します．［Add Source...］ボタンをクリックしてファイルを指定してください．sim.vはシミュレーション記述用ファイルですが，今回は必要ないので除外します．また，制約条件ファイルであるtop.ucfの中でピン割り当てを指定します．そして，［開く］ボタンを押すと，各ファイルについてChoose Source Typeダイアログが開きます．ここでは，すべて「Verilog Design File」を選択してください．これでファイルが登録されたので，［次へ］ボタンをクリックします．すると，5ページ目の作成プロジェクトの説明が表示されます．内容を確認して，［完了］ボタンを押します．

最後に，ucfの適用対象ソース・ファイルについて問い合わせがありますが，ここでは「top」を指定してください．



図A-1　Project Navigatorの起動とNew Projectの選択



図A-3　プロジェクト生成後の画面

すると新プロジェクトが作成され，**図A-3**のようなウィンドウとなります．画面左側でソース・ファイルの選択や論理合成，配置配線などのコマンドを指定すると，右側に選択したソース・ファイルが表示されます．また，下側には各種メッセージが表示されます．

さて，実際に論理合成を実行してみましょう．**図A-4**に示すように，画面左側のSources in Projectウィンドウにある「top（top.v）」を選択し，Process for Sourceウィンドウの「Generate Programming File」をダブルクリックします．すると，topモジュール以下のHDLファイルの論理合成が開始されます．数分の後，画面下のメッセージ欄に「Completed process "Generate Programming File"」と表示され，論理合成が完了したことが通知されます．

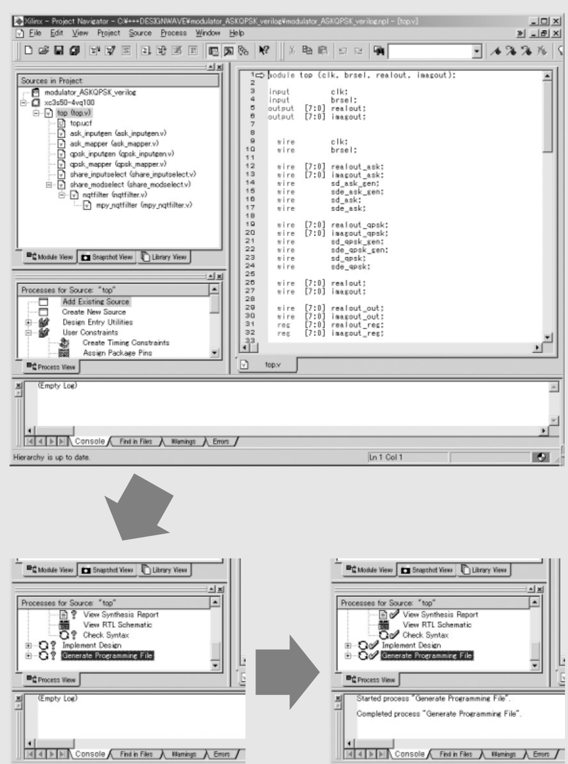
コンソールにはさまざまな情報が表示されます．スクロール・バーを動かすと，論理合成ツールのレポートを見ることができます．例えば，今回の例では動作可能周波数が「55.602MHz」，使用スライス（Slice）数が「702/768（91%）」となっています．

● PROM用データを作成する

次に，PROM用データの作成方法について説明します．

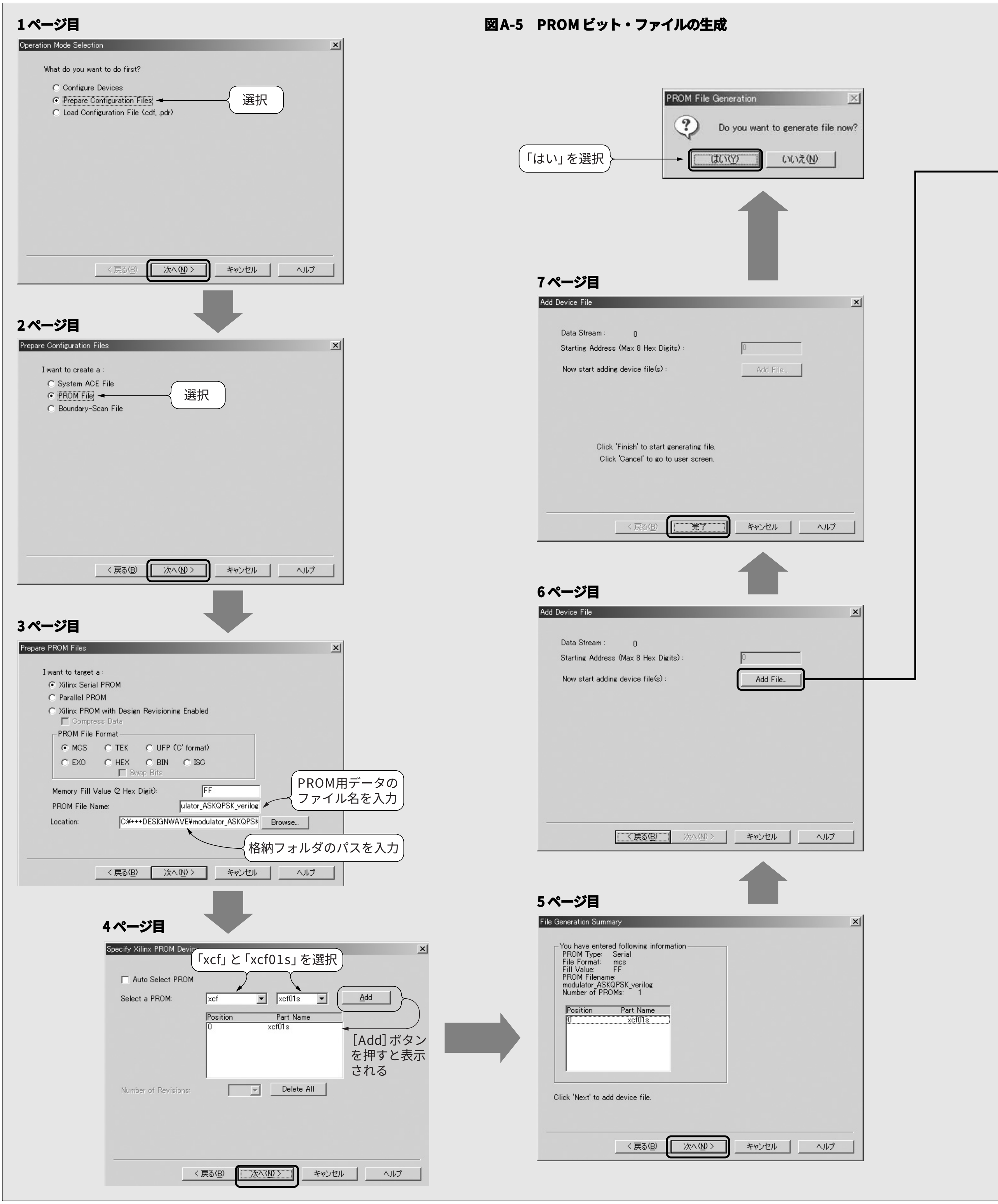
Windowsのスタート・メニューから「プログラム」→「Xilinx ISE 6」→「アクセサリ」→「iMPACT」を選択します．すると，**図A-5**の1ページ目のようなウィンドウが表示されます．PROM用コンフィグレーション・ファイルを作成するため，「Prepare Configuration File」を選択し，次に進みます．2ページ目では，「PROM File」を選択します．

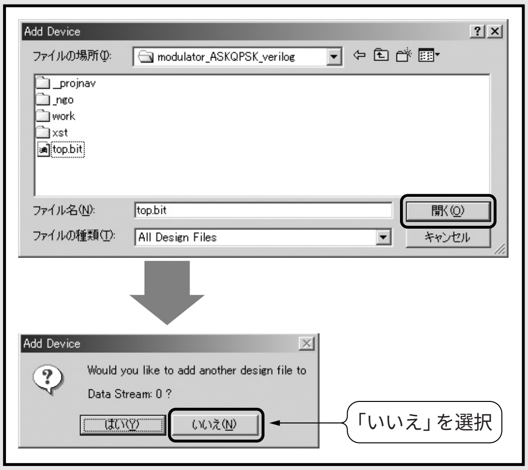
続く3ページ目では，PROM用データのファイル名をPROM File Name欄に，またその格納フォルダをLocation欄に入力します．4ページ目では，対象とするPROMを指定します．今回は，FPGA基板

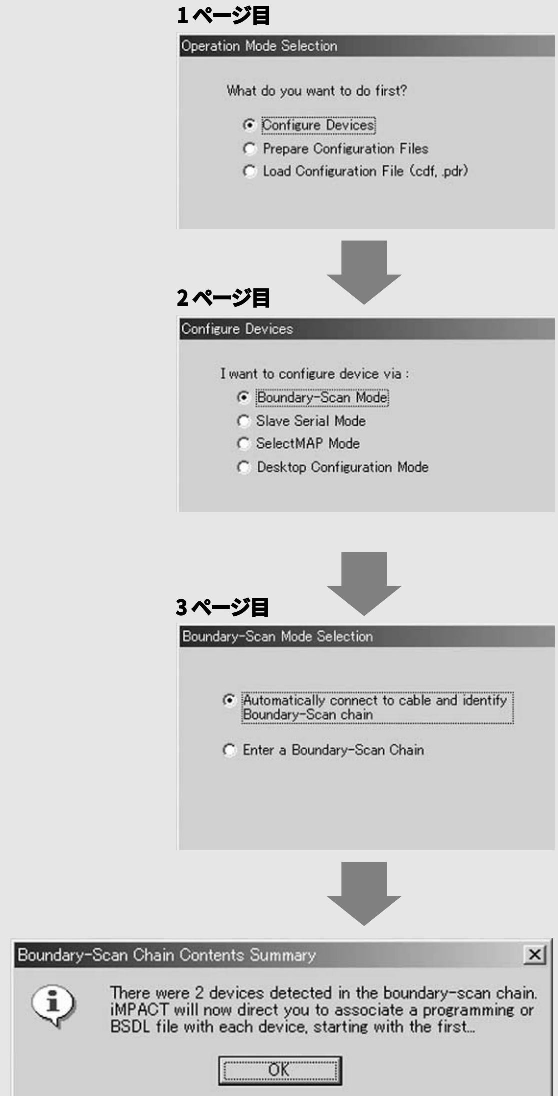


図A-4　論理合成を開始する

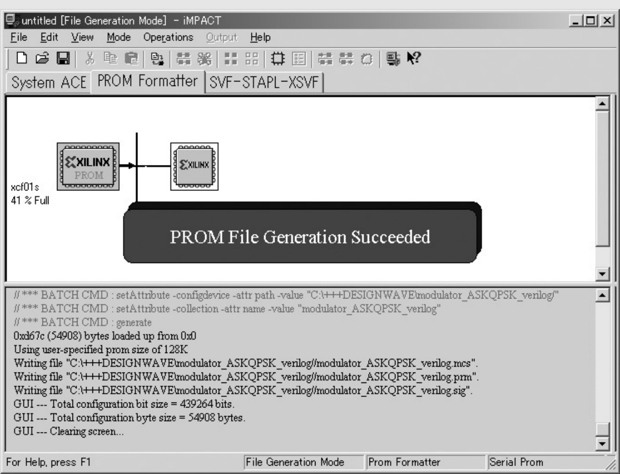
図A-5　PROMビット・ファイルの生成







図A-7　PROMへの書き込み設定



図A-6　PROMファイル生成画面



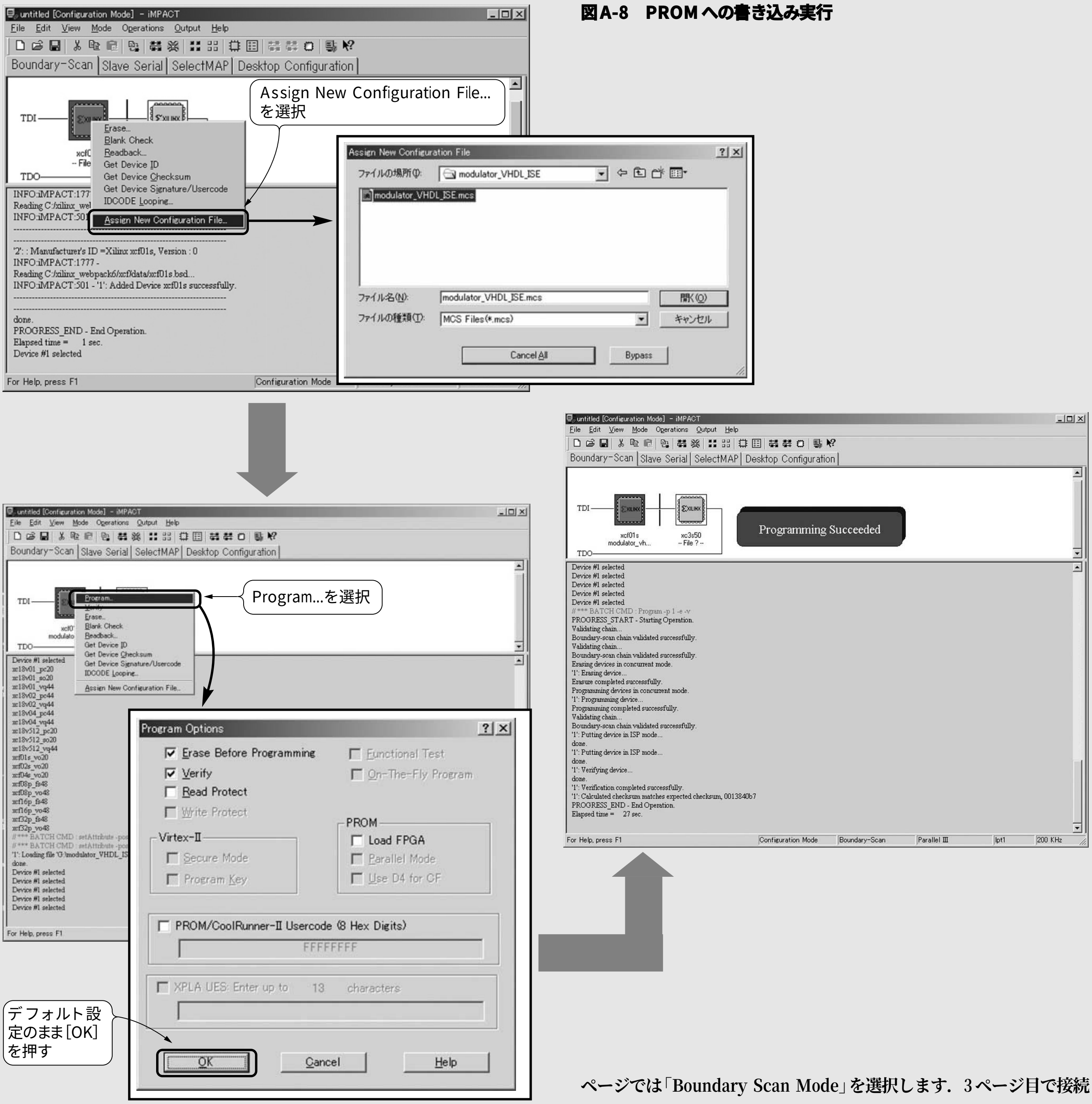
写真A-1　ダウンロード・ケーブル(Parallel Cable IV)

に搭載したROMである「xcf」，「xcf01s」を指定し［Add］ボタンを押してください．そして，［次へ］ボタンをクリックします．指定した内容が表示されるので，確認してから，さらに［次へ］ボタンをクリックします．

6ページ目のウィンドウでは，PROMデータのもとになるビット・ファイルを指定します．［Add File...］ボタンをクリックして，先ほど論理合成ツールで作成したビット・ファイル(top.bit)を指定します．ほかにもビット・ファイルを追加するかどうかを尋ねるダイアログが表示されますが，今回PROM化するのは一つの設計だけなので，［いいえ］ボタンをクリックします．

7ページ目で［完了］ボタンをクリックします．すると，PROM File Generationダイアログが開くので，直ちにPROMデータを作成するため，［はい］ボタンをクリックします．**図A-6**のようなウィンドウが表示されたら成功です．これによって，先ほど指定したフォルダ内にPROMコンフィグレーション用ファイル(mcs)が作成されました(な

図A-8　PROMへの書き込み実行



お，今回設計したπ/4シフトQPSK変調の送信機，およびASK変調とπ/4シフトQPSK変調を切り替えられる送信機のROMファイルも，本誌付属のDVD-ROMに収録されている）．

● PROMにデータを書き込む

最後に，実際にFPGAボード上のPROMへの書き込みを行います．今回はXilinx社のダウンロード・ケーブル「Xilinx Parallel Cable IV」を利用しました（**写真A-1**）．FPGAボードとリード・ワイヤで接続した後，上述の手順で「iMPACT」を起動します（**図A-7**）．

1ページ目では，今回は「Configure Devices」を選択します．次のページでは「Boundary Scan Mode」を選択します．3ページ目で接続を自動検出させるため，上側の項目を選択します．すると，2デバイス，すなわちPROMとFPGAが検出されました．

ここで，PROMに書き込むため，**図A-8**のように左側の「xcf01s」をマウスで右クリックし，「Assign New Configuration File...」を選択します．Assign New Configuration Fileダイアログが開くので，先ほど作成したPROM用ファイル（.mcsファイル）の名まえを指定します．再度，左側の「xcf01s」を右クリックして，今度は「Program」を指定します．Program Optionsダイアログが表示されますが，デフォルトのままで［OK］ボタンをクリックします．約1分後，「Programming Succeeded」と表示されれば成功です．

これで，FPGAボードの電源投入時に自動的にPROMからFPGA用のコンフィグレーション・データがロードされ，処理が開始されます．